Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

FORMAL ANALYSIS OF AIR TRAFFIC MANAGEMENT SYSTEMS: THE CASE OF CONFLICT RESOLUTION AND RECOVERY

Ricky Butler
Jeffrey Maddalon

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681, U.S.A.

Alfons Geser
César Muñoz

National Institute of Aerospace
Hampton, VA 23666, U.S.A.

ABSTRACT

New air traffic management concepts distribute the responsibility for traffic separation among the several actors of the aerospace system. As a consequence, these concepts move the safety risk from human controllers to the on-board software and hardware systems. One example of the new kind of distributed systems is air traffic conflict detection and resolution. Traditional methods for safety analysis such as human-in-the-loop simulations, testing, and flight experiments may not be sufficient in this highly distributed system: the set of possible scenarios is too large to have a reasonable coverage. This paper proposes a paradigm shift for the safety analysis of avionics systems where formal methods drive the development of critical systems. As a case study of this approach, we report the mechanical verification of an algorithm for air traffic conflict resolution and recovery.

1 INTRODUCTION

Air Traffic Management (ATM) has two competing objectives: maximize the efficiency of the airspace system and provide a smooth and safe flow of traffic. One of the most critical responsibilities of an ATM system is to maintain traffic separation. Today, this responsibility resides in a central authority, e.g., an Air Traffic Service Provider (ATSP). The ATSP monitors the airspace and issues clearances that are expected to be followed by the aircraft. Efficiency is often sacrificed for safety and there is little room for user preferences. Novel approaches to ATM, e.g., Distributed Air-Ground Traffic Management (DAG/TM) (NASA 1999), Free-flight (RTCA 1995, Hoekstra et al. 2000), address efficiency problems of the current airspace system by distributing the responsibility for traffic separation among all the aircraft in the airspace. In these approaches, on-board hardware and air traffic management software provide surveillance, alert for possible loss of separation, and advise corrective maneuvers.

On-board conflict detection and resolution (CD&R) systems are critical components of new ATM concepts. Since no human controller checks the output, the fundamental responsibility for air traffic separation resides on distributed CD&R systems. Safety analysis of a CD&R algorithm amounts to showing that for every possible scenario, conflicts are detected and effectively solved. Traditionally, this is done via extensive testing, human-in-the-loop simulations, and flight experiments. We argue that the traditional techniques are not sufficient in this new distributed environment. Human-in-the loop simulations, like all simulations, can only describe phenomena that they have specifically modeled. In addition, simulation results can be corrupted by an unintentional bias in selecting scenarios for test. Flight experiments are too expensive to obtain a significant number of results. Worst of all, even when discretized, the set of possible scenarios is too large to obtain a reasonable coverage with testing, simulation, and experimentation.

In this paper we propose a formal approach to safety analysis of future ATM systems. As an illustration of the first step in using this approach, we report the mechanical verification of an algorithm for air traffic conflict resolution and recovery, called RR3D (Geser et al. 2002). The RR3D algorithm adds arrival time constraints to a state-based geometric CD&R algorithm (Dowek et al. 2001). It may be seen as a building block for strategic conflict resolution. We have formally verified RR3D in the verification system PVS (Owre et al. 1992). In our view, this verification is an important step toward a new approach for safety analysis of air traffic management systems, where formal methods drive the development and validation of critical systems.

The rest of the paper is organized as follows. Section 2 discusses the rationale for a *formal* safety analysis methodology. Section 3 presents a short overview to conflict detection and resolution modeling techniques. Section 4 introduces the resolution and recovery algo-

rithm RR3D. RR3D serves as a case study for our formal approach to safety analysis in Section 5. Section 6 summarizes our work and discuss future research directions.

2 WHY FORMAL SAFETY ANALYSIS

Digital avionics systems have been used since the early seventies. A fly-by-wire aircraft such as the Boeing 777 employs safety-critical software in its flight control computers. This type of software is largely derived from control theory based on rigorous mathematical methods that provide assurance of key properties such as stability. Moreover, the basic stability of the aircraft provides protection from occasional glitches in the control software.

On the ground side, most of the software associated with ATM is packaged into decision support tools for air traffic controllers, e.g., Center TRACON Automation System (CTAS) (Sanford et al. 1993). This software provides information to controllers in a convenient format to aid them managing the trajectories of the aircraft in their sector. The failure of this software is mitigated by human intelligence that has many sources of information about the aircraft under ATM control including analog display of radar data. Consequently, the safety risk resides primarily in the human controllers. The main question to be asked about such software is whether the software helps the controllers achieve their operational goals. This question is best answered by statistically designed and human factors oriented experiments.

Future ATM concepts under development will utilize software in ways that are fundamentally different from the past. Many of these concepts move the safety risk directly into executing software. A near-term example of this is the ICAO's (International Civil Aviation Organization) Required Navigation Performance (RNP) initiative. RNP extends the capabilities of modern airplanes by providing more accurate and precise navigation capability leading to more flexible airspace routes and procedures in both visual and instrument conditions. Although the RNP system will provide greater accuracy, it will necessarily rely on more sophisticated on-board software and external infrastructure such as Global Positioning System (GPS) and their associated systems such as the Wide Area Augmentation System (WAAS). In these future ATM systems the safety risk migrates from radar and controllers to on-board software and critical technologies, such as GPS, that are also dependent upon software systems. This software consequently has a new safety role because no human checks its validity. Hence, it is reasonable to re-examine the methods by which we determine that software is correct and reliable.

The safety analysis of air traffic management systems cannot be accomplished using simulation and experimentation alone. To verify that a piece of software is correct, one must ensure that there are no reachable unsafe states. Unfortunately, the state space of complex systems is astronomically large. The input space alone must cover the 3-D airspace in the vicinity of an aircraft and all possible pilot inputs. Even if these are discretized, the number of test cases that must be examined to cover the input domain would require millions of years of experimentation. Extensive simulation can only establish that *a few* states, compared with the enormous set of possible states, are safe. From there, it is unrealistic to infer that *all* states, or even that *most* states, are also safe. A complete coverage of the system set of states and the rigorous analysis of its safety properties is only possible through *Formal Methods*.

Some have argued that since there are many unpredictable elements in flight management, e.g., changing weather, system failures, human errors, etc., it is impossible to achieve any guarantee about the behavior of ATM algorithms in a systems context. They then conclude that a formal analysis of an ATM system is not useful. Although it is not possible to issue an absolute guarantee under all possible eventualities that an algorithm will produce a successful outcome, formal techniques can guarantee that an algorithm is correct for all possible scenarios *under well-defined assumptions*. As we will explain later, the explicit set of hypotheses under which safety properties are valid is a by-product of formal verification. In this paper we argue that formal methods is an essential step in the validation process of avionics systems.

In engineering when one encounters an extremely complex and unpredictable environment, one seeks to bring mathematical rigor to as much of the system's domain as possible. This is done to minimize the uncertainty in the system. One way to view formal analysis is that all systems have a *behavior* that is dependent on *assumptions* about the environment in which the system operates and the *logic* contained within the system. If the behavior of the system is incorrect then it must be the case that either the assumptions or the logic are incorrect. Formal verification ensures that the expected behavior, i.e., the system requirements, matches the logic, as long as the assumptions are valid. If a formally verified system fails, then it *must* be the case that the assumptions are not valid. Formal verification does not simply produce a list of assumptions, it also provides a framework where experts can uncover assumptions. It is critical that the assumptions on which the system was built are validated. Validating assumptions can only be accomplished by human inspection, flight experiments,

and simulations. Therefore, extensive simulations must still be conducted to establish that the operational procedures that govern the new airspace concept are adequate to sustain the *assumptions* that go into the formal analysis of the software algorithms. And flight experiments are performed to validate the assumptions of the simulations. However, the idea that a flight experiment can demonstrate the safety of an air traffic management concept must be rejected. The number of input cases covered in any flight experiment is so minuscule that its usefulness for this purpose is essentially nil. Nevertheless, a flight experiment provides a critical capability in that it can discover shortcomings and errors in the assumptions that form the foundation for the analysis. When problems are discovered here, the analysis must be adjusted to reflect the more realistic characteristics of the environment or the operational procedures must be modified in order to rule-out the discovered problem area.

A credible safety case for an advanced ATM system will be a massive undertaking. The following is only a rudimentary list of some of the key characteristics of a comprehensive safety case.

- All of the requirements for safety must be captured and expressed in a rigorous manner.
- Verifiable algorithms and designs must be used whose behavior is fully explicated via mathematical theorems.
- The software implementations have been developed in accordance with certification standards, such as DO-178B, and shown to be faithful refinements of the formally verified algorithms using code-level verification.
- The operating system on which the software implementation executes must provide guarantees of integrity and performance.
- The operational procedures have been shown to be complete and safe and extensively simulated.
- All of the assumptions of the formal analysis have been subjected to extensive investigation through simulation and flight experimentation.
- The probability of failure (due to physical faults) of critical components and in the infrastructure systems must be shown to meet strict reliability requirements on the order of 10^{-9}.
- The adequacy of the fault-tolerance strategies must be accomplished using fault-trees and Markovian analysis as well as laboratory experimentation.
- The pilot/controller workload associated with the advanced systems must be shown to be reasonable via simulated and flight experiments.
- All of the traditional environmental simulation and experimentation, such as DO-160, must also be performed.

We believe that the existing incremental approach to system safety will be inadequate to convince regulatory agencies, such as the Federal Aviation Administration (FAA) in the US, that future ATM systems that rely on complex distributed software implementations are certifiably safe. We believe that safety cases built on the foundation of provably correct algorithms and designs is the only viable approach for future ATM systems.

As a first step toward a safety case of an advanced ATM concept, we report in this paper the mechanical verification of an algorithm for conflict resolution and recovery, called RR3D (Geser et al. 2002). The original presentation of that algorithm contains a hand-written proof of its correctness. Although, in essence, the algorithm is correct, the mechanical verification revealed missing assumptions and a few errors in the hand-written proof. This supports our belief that mechanical verification is valuable even when the system has been diligently analyzed. Without a mechanical proof, it is almost impossible to find such kind of errors. A missing assumption, for example, could result in a fatal error in a real implementation of RR3D.

Since the RR3D algorithm has been formally verified, we are confident that it is logically correct. Nevertheless, this algorithm must be translated into a machine-executable language, such as Ada or C, and interact with the external environment. This will necessitate several more steps of logical design each potentially vulnerable to errors being introduced. There are many issues that must be addressed as this is done:

1. The algorithm operates over the real numbers not floating point numbers. The executable code must deal with overflow, underflow, and all the usual numerical problems.
2. The algorithm assumes no errors are present in the state data of the aircraft involved. But even the best sensors provide only approximate values and so the effect of this error must be handled. Furthermore, the system must be able to handle some number of failures, i.e., it must be fault-tolerant, so these design refinements must be rigorously examined as well.
3. The algorithm operates in a real-time environment, so one must establish that the system on which the algorithm executes has sufficient CPU time (under

all possible scenarios) to complete the RR3D algorithm.

This process of design refinement can itself be captured in a sequence of successfully more complete formal models finally resulting in an implementation or a detailed specification from which an implementation can be synthesized. Each of these formal models can be shown to satisfy all the properties of the higher model. This process is usually referred to as *design proof* and the final verification that carries one down to the implementation code is called *code verification*. If the last step is accomplished using synthesis, the code that implements the synthesizer itself must be verified or its output validated against the detailed design. It should be pointed out, that our work on RRD3 has only accomplished the first step, namely the top-level proof that the mathematical algorithm meets its specified properties. Future work will look at more of these system level issues.

3 CONFLICT DETECTION, RESOLUTION, AND RECOVERY

CD&R algorithms are designed to warn about potential loss of air traffic separation and output avoidance maneuvers to be flown by the aircraft.

There is a wide variety of approaches to CD&R because there are different ways to (1) predict the future trajectories, (2) define what constitutes close proximity of trajectories, (3) calculate the resolution trajectories, and (4) gain assurance about the safety and effectiveness of the algorithms. Algorithms also differ in the domain of application: (1) how far ahead in time should a conflict be detected, (2) whether the algorithm deals with only 2 conflicts at a time or handles multiple simultaneous conflicts, and (3) the amount of coordination and communication needed to implement the algorithm. Kuchar and Yang (2000) lists several CD&R modeling methods and proposes a taxonomy to classify them.

Furthermore, in the recent years, new approaches for CD&R have been proposed that use non-standard programming techniques such as genetic algorithms (Durand et al. 1996, Granger et al. 2001, McDonald and Vivona 2000), neural networks (Durand et al. 2000), game theory (Tomlin et al. 1998), graph theory (Chiang et al. 1997), and semi-definite programming (Frazzoli et al. 2001). Given the computational complexity of some of these techniques, they usually require costly time and space discretizations. In contrast to these approaches, the geometric approach (Eby 1994, Hoekstra et al. 2000, Bilimoria 2000, Dowek et al. 2001) is based on standard and well-understood analytical techniques. In Kuchar & Yang's taxonomy, the geometric modeling correspond to nominal trajectories with either optimized or force field resolutions. Nominal trajectories are linear projections of the current position and velocity vectors. The conflict resolution problem is then expressed as a set of polynomial equations that are solved using classical analytical techniques. Since linear projections produce prediction errors that are negligible for short look-ahead times, this approach is also referred to as *tactical*. For large look-ahead times a more strategic approach that looks at the pilot intent information, e.g., flight plan, is in order. While tactical approaches have well-understood geometric descriptions that allow for efficient and clear algorithms, they may fall short on pilots' expectations (Wing et al. 2001).

Resolution *and recovery* algorithms, called resolution with arrival time constraints in (Bilimoria and Lee 2002), generate, in addition to the avoidance maneuver, return trajectories that bring an aircraft back to its nominal path.

Figure 1 illustrates the environment where conflict resolution and recovery takes place in an abstract distributed ATM system. On-board measurement devices capture the current state of the aircraft and broadcast this information to all the aircraft in the same sector. When the conflict detection module detects a potential conflict within a look-ahead time, the resolution and recovery module computes a list of escape and recovery maneuvers. The choice of maneuvers is displayed at the cockpit interface for pilot selection or it may be input to a navigation system that automatically selects the optimal maneuver among the choice.

4 RR3D

In RR3D, aircraft are represented by a kinematic particle model with the center of gravity as the coordinate point. Furthermore, trajectories are assumed to be composed of linear segments: speed is constant within a segment and from one segment to another acceleration is instantaneous. RR3D resolves conflicts in pairwise fashion where the traffic aircraft (also called intruder) is surrounded by a cylindrical protected zone P of diameter $2D$ and height $2H$, where D is the required horizontal separation and H is the required vertical separation. A *conflict* is an intrusion of the ownship in the traffic's protected zone. RR3D computes conflict-free escape and recovery maneuvers that are tangential to the intruder's protected zone.

For simplicity, we chose a relative coordinate system where the intruder aircraft is fixed at the origin. RR3D has the following inputs:

- Relative position $\vec{s}$ of ownship with respect to intruder.

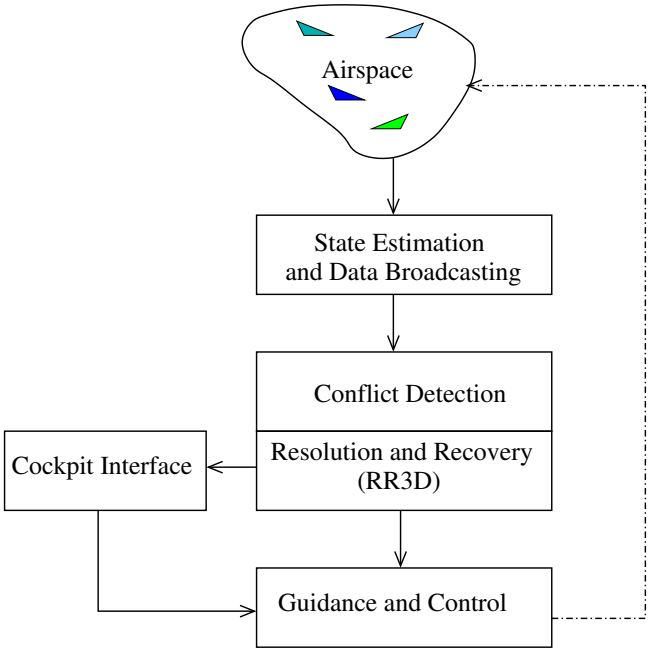

Figure 1: Distributed ATM System

- Velocity vector of ownship $\vec{v_o}$.
- Velocity vector of intruder aircraft $\vec{v_i}$.
- Arrival time t'' at a relative target point s'', which is defined as
$$\vec{s''} = \vec{s} + t''\vec{v},$$
where $\vec{v} = \vec{v_o} - \vec{v_i}$.

RR3D outputs a choice of escape and recovery maneuvers for the ownship, i.e., triples $(\vec{v_o'}, t', \vec{v_o''})$ where $\vec{v_o'}$ is the escape velocity vector, t' is the time of turn, and $\vec{v_o''}$ is the recovery velocity vector. Figure 2 illustrates RR3D's functionality for a single output.

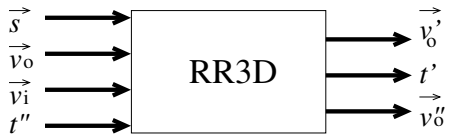

Figure 2: RR3D: Input/Outputs

Escape and recovery maneuvers are constrained in such a way that both $\vec{v_o'}$ and $\vec{v_o''}$ satisfy one of the following conditions:

1. *Change of vertical speed only.* The ownship's vertical speed may change but neither its heading nor its ground speed may change. Formally,
$$v'_{ox} = v_{ox} = v''_{ox},\ v'_{oy} = v_{oy} = v''_{oy}. \qquad (1)$$

2. *Change of ground speed only.* The ownship's ground speed may change but neither its heading nor its vertical speed may change. Formally, there are $k > 0, j > 0$ such that
$$\begin{array}{c} v'_{ox} = kv_{ox},\ v'_{oy} = kv_{oy},\ v'_{oz} = v_{oz}, \\ v''_{ox} = jv_{ox},\ v''_{oy} = jv_{oy},\ v''_{oz} = v_{oz}. \end{array} \qquad (2)$$

3. *Change of heading.* The ownship's heading and ground speed may change. In the two dimensional projection, the escape course and the recovery course (each in absolute coordinates) form a triangle. By the triangle inequality, the escape course and the recovery course together are longer than the original course. To arrive at the target point at time t'', the ownship has to compensate the longer way by a greater average ground speed as opposed to its original ground speed. Hence, maneuvers where only heading changes are allowed cannot reach the target point in time. In this case, we propose a change of heading combined with a change of ground speed at time t'. For the escape step, the ownship's heading may change, but neither its ground speed nor its vertical speed; for the recovery step in addition to a heading change, one must allow for a change of ground speed as well. Formally,
$$\begin{array}{c} v'^2_{ox} + v'^2_{oy} = v^2_{ox} + v^2_{oy}, \\ v_{oz} = v'_{oz} = v''_{oz}. \end{array} \qquad (3)$$

Furthermore, we require that the escape and recovery courses are tangential to the lateral surface of the protected zone. Tangential courses solve a predicted conflict in an optimal way. They require the least effort to correct the original trajectory such that the ownship arrives at the next trajectory change point at the scheduled time while maintaining separation. We also request that the turn time t' be constrained by $0 < t' < t''$. Original, escape, and recovery courses are illustrated in Figure 3.

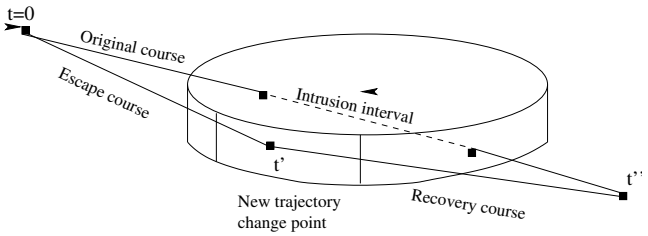

Figure 3: RR3D: Graphically

5 FORMAL VERIFICATION OF RR3D

A unique feature of the RR3D algorithm is that its functional behavior has been mathematically analyzed. More specifically, Geser et al. 2002 present a rigorous proof of the following property that we call RR3D *correctness*. Given that

- aircraft are not in conflict at either the initial point nor at the target point

$$\begin{array}{c} s_x^2 + s_y^2 > D^2 \quad \text{or} \quad s_z^2 > H^2, \\ s''^2_x + s''^2_y > D^2 \quad \text{or} \quad s''^2_z > H^2, \end{array} \tag{4}$$

- aircraft are in predicted conflict: there is a time $0 < t < t''$ such that

$$\begin{array}{c} (s_x + tv_x)^2 + (s_y + tv_y)^2 < D^2, \\ (s_z + tv_z)^2 < H^2, \end{array} \tag{5}$$

the following propositions hold:

- Escape course maintains separation. Let $\vec{v'} = \vec{v'_o} - \vec{v_i}$; then for all times $0 \le t \le t'$

$$\vec{s} + t\vec{v'} \notin P. \tag{6}$$

- Recovery course maintains separation. Let $\vec{v''} = \vec{v''_o} - \vec{v_i}$; then for all times $t' \le t \le t''$

$$\vec{s} + t'\vec{v'} + (t - t')\vec{v''} \notin P. \tag{7}$$

- Arrival time constraint is respected. Formally,

$$\vec{s} + t'\vec{v'} + (t'' - t')\vec{v''} = \vec{s''}. \tag{8}$$

The formal verification essentially follows the handwritten proofs in Geser et al. 2002. However, the formal effort revealed a few assumptions that were missing and some logical errors in the original argument. This is not surprising. By formalizing every detail of the correctness argument, mechanical verification enables the discovery of errors that otherwise would be almost impossible to find. The PVS proofs of correctness and of satisfaction of the chosen constraints are complete. The formal specification of the algorithm in PVS, including 431 claims, is about 3K lines. The correctness proof for these claims is about 19K lines in size. This development is available as a PVS dump at `<http://research.nianet.org/fm-at-nia>`.

Geser et al. 2002 describe the RR3D algorithm as a set of solutions to polynomial equations that satisfy one of the constraints (formulas 1, 2, or 3), the initial assumptions (formulas 4 and 5), and the correctness property (formulas 6, 7, and 8). The solutions are categorized according to the part of the surface of P that is touched during the escape and recovery courses. In particular, the ownship may touch the cylinder either at its lateral boundary, then we speak of a *line case*, or at its top or bottom disks, then we speak of a *circle case*. If only the disks are touched then one disk may be touched once or twice, or both disks may be touched once each. For instance, Figure 3 illustrates a line-line case, i.e., both escape and recovery courses are line cases.

The combinations of these sub-cases produce a large number of resolution and recovery maneuvers. The RR3D algorithm evaluates each case. If a suitable maneuver is not possible for a particular case, then the algorithm reports no maneuver for that case. The algorithm collects all solutions and produces a list of escape and recovery maneuvers. The interesting part of the formal verification is to show that given a case where a solution is generated for particular constraint (formulas 1, 2, and 3), if the initial state satisfies formulas 4 and 5, the solution satisfies formulas 6, 7, and 8.

The basic problem we encountered during the formal verification is that of managing complexity. We address this problem by stating, proving, and reusing lemmas about common parts or aspects of the design. For instance, after a preliminary analysis of the problem, we realize that in all the cases, correctness is achieved by combination of the following criteria:

```
line_case_correctness : THEOREM
    hor_sep?(s) AND
    hor_pass?(-1,s,v) AND hor_pass?(1,s,v)
  IMPLIES separation?(s,v)

circle_case_correctness : THEOREM
    hor_sep?(s) AND hor_pass?(eps,s,v) AND
    vert_sep?(s) AND vert_pass?(-eps,s,v)
  IMPLIES separation?(s,v)
```

Here, the propositions `hor_sep?(s)` and `vert_sep?(s)` denote the inequations $s_x^2 + s_y^2 \ge D^2$ and $s_z^2 \ge H^2$, respectively. They state that the point s is horizontally or vertically separated from the intruder. The propositions `hor_pass?(`ε`,s,v)` and `vert_pass?(`ε`,s,v)` denote the inequations $\varepsilon s_z v_z \ge 0$ and $\varepsilon(s_x v_x + s_y v_y) \ge 0$, respectively. They state that the velocity vector v has a horizontal or vertical component in the direction of s (for $\varepsilon = 1$) or in the opposite direction (for $\varepsilon = -1$). If `hor_pass?(`ε`,s,v)` holds for both $\varepsilon = 1$ and $\varepsilon = -1$ then the horizontal projections of s and v are orthogonal. In this case, s is the closest approach point to the intruder.

The `line_case_correctness` theorem states that the moving point $s + tv$ is separated at any time t, provided that the point s is horizontally separated and the

horizontal projections of v and s are orthogonal. Intuitively, v points to a tangent direction at radius vector s. The `circle_case_correctness` theorem states that the moving point $s+tv$ is separated at any time t, provided that the point s is both horizontally and vertically separated and the inner product of the horizontal projections of v and s has a sign opposite to the inner product of the vertical projections of v and s. Intuitively, s is the point where horizontal separation ends and vertical separation starts, or vice versa. We use each theorem with s instantiated with the touch point.

Let us study the line-line case where only the horizontal speed is changed. For ground speed change, the ownship's new velocity vector during escape course is $v_o' = (kv_{ox}, kv_{oy}, v_{oz})$ where the real number $k > 0$ denotes the magnitude of speed change. There is a similar formula for the recovery course and a magnitude $j > 0$. The case routine first determines k and so the solution. It then checks the solution for eligibility. The following lemma in PVS forms the basis of the separation proof of the escape course, which is a line case:

```
gs_l_esc_sep: LEMMA
  ground_speed_change?(k,vo,vi,v') AND
  v = vo - vi AND
  hor_strict_sep?(s) AND
  hor_move?(vo) AND
  pred_conflict?(s,v,t'') AND
  kappa_defined?(s,vo,vi) AND
  k = kappa(eps,s,vo,vi) IMPLIES
    separation?(s,v')
```

This lemma states that the ownship's relative movement $s+tv'$ during the escape course maintains separation to the intruder for all time t (`separation?(s,v')`), provided that v_o' is a ground speed change from v_o (`ground_speed_change?(k,vo,vi,v')`), i.e., $v_o' = v' - v_i = (kv_{ox}, kv_{oy}, v_{oz})$ by a factor of k where k is given by `kappa(eps,s,vo,vi)`; the starting point s is horizontally separated and not at the boundary (`hor_strict_sep?(s)`); the ownship's ground speed is not zero (`hor_move?(vo)`); and a conflict is predicted for the original trajectory (`pred_conflict?(s,v,t'')`).

In the proof of the lemma, we use the `line_case_correctness` theorem, instantiated by the touch point $s + \tau(s, v')$, where $\tau(s, v')$ is the time of closest approach to the protected zone for the escape course. The theorem yields `separation?`$(\mathtt{s} + \tau(\mathtt{s}, \mathtt{v}'), \mathtt{v}')$ which is easily shown equivalent to the claim `separation?(s,v')`. This leaves us to discharge the assumptions $\Delta(s, v') = 0$ and `hor_sep?`$(s + \tau(s, v'))$. The equality $\Delta(s, v') = 0$ indicates a tangent to the infinite cylinder through point s. From the premise `pred_conflict?(s,v)` we can infer `hor_move?(v)`, i.e., that v has a non-zero horizontal projection. Then `hor_sep?`$(s + \tau(s, v'))$ follows from `hor_move?(v)` and $\Delta(s, v') = 0$. In order to show $\Delta(s, v') = 0$, we first show $\Delta(s, v) > 0$ which follows from `pred_conflict?(s,v)`, and show that then `kappa(eps,s,vo,vi)` is a solution of the quadratic equation $\Delta(s, v') = 0$. The inequality $\Delta(s, v) > 0$ indicates that there are two intersections of the movement $s + tv$ with the lateral boundary of P. This is the case with a predicted conflict.

There is a similar lemma for line-recovery, the line case of the recovery course. The factors k and j together determine the time t' by the timeliness goal. Another lemma states that if we have a predicted conflict then the "ground-speed/line/line" case routine of the algorithm provides all premises of lemma `gs_l_esc_sep`. Put together they form the correctness proof of the ground-speed/line/line case of RR3D. By exchanging the recovery course with a circle-recovery case we obtain the ground-speed/line/circle case, and so forth.

6 CONCLUSION

In this paper, we argue for a formal approach to the development of safe Air Traffic Management (ATM) systems. We also report the formal verification of a critical component of a distributed ATM concept: an air traffic resolution and recovery algorithm.

Formal verification provides a systematic way to identify and reduce the unpredictability. By a formal verification, the designer documents all assumptions unambiguously, and demonstrates full comprehension of the verified component including the interface with neighbor components. This helps the designer to make necessary adjustments to the components that do not quite fit the interface. So we claim that having a set of algorithms whose behavior is fully understood under explicitly stated assumptions greatly aids the designer of ATM operational systems. Not only is the designer liberated from having to think about contingency plans for failures of the algorithm, but by knowing the assumptions built into the algorithm, the designer has explicit knowledge about where to focus attention to produce a robust and safe operational concept. In this approach, human-in-the-loop simulation and expensive flight experiments are used to validate assumptions made during the formal verification. This is major shift from traditional approaches where testing and simulation drive the safety validation and certification of avionics systems.

We should note that a proof of correctness of an algorithm does not guarantee a fault-free system. This is because the algorithm implicitly makes idealized assumptions. The verification of a system implementa-

tion must therefore provide a proof that the algorithm is faithfully implemented. This includes issues such as floating point overflow and underflow, rounding errors, validity of input data, real-time deadlines on execution, communication flaws, etc. Furthermore, at the system design level additional algorithms are introduced to handle inter-aircraft communications (e.g. ADS-B), to detect and mask faulty input data, to format output data for pilot displays, to schedule the execution, to coordinate with other systems such as flight planners, etc. These algorithms, too, must be shown to satisfy critical safety properties.

The verification of a resolution and recovery algorithm is only a first step toward the system verification of an ATM system. As a next step the RR3D algorithm may be refined into a high-level design, which is then translated into a programming language. This step will be accompanied by formal proofs of the faithfulness of the transitions. An ATM system that integrates an implementation of RR3D will be formally supported by several layers of abstraction as illustrated in Figure 4.

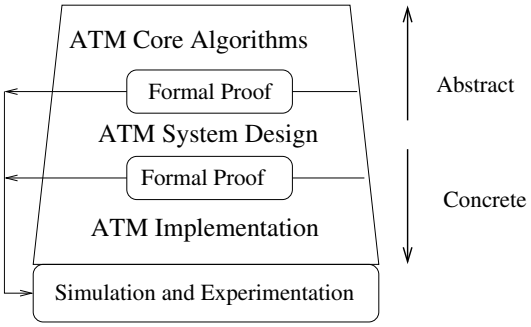

Figure 4: System Verification

Finally, we enumerate some issues related to system verification that we are currently looking into or planning to do so in the near future.

- **Strategic CD&R**. RR3D is a state-based CD&R algorithm with minimal intent information. It propagates an aircraft trajectory based on its current location, velocity vector, and arrival time constraint. The arrival time constraint makes RR3D suitable for strategic CD&R. Indeed, Geser and Muñoz 2002 describe an algorithm that incorporates RR3D into a conflict-free flight planner. The correctness of the flight planner is based on the correctness of RR3D. The resolution and recovery algorithm effectively helped us to decompose the complexity of both the flight planner *and*, more importantly, its correctness proof.

- **Geodesic Coordinates**. As most geometric CD&R algorithms, RR3D is presented in a Cartesian coordinate system assuming a flat earth. On top of RR3D, we have developed an interface module that converts from geodesic coordinates to Cartesian system that minimizes errors due to the flat earth assumption. The formalization and correctness proof of the coordinate transformation is in progress.

- **Floating Point Errors**. The verification of RR3D assumes exact real arithmetic. In contrast, usual programming languages provide floating point arithmetic. It is well-known that floating point numbers violate some elementary properties of real numbers. An interval analysis of RR3D that considers floating point errors, underflows, and overflows will complement a preliminary work on refinement of abstract algorithms into real-life programming languages.

REFERENCES

Bilimoria, K. 2000, August. A geometric optimization approach to aircraft conflict resolution. In *Guidance, Navigation, and Control Conference*, Volume AIAA 2000-4265. Denver, CO.

Bilimoria, K., and H. Lee. 2002, August. Aircraft conflict resolution with an arrival time constraint. In *Guidance, Navigation, and Control Conference*, Volume AIAA 2002-4444. Monterey, CA.

Chiang, Y.-J., J. Klosowsky, C. Lee, and J. Mitchell. 1997. Geometric algorithms for conflict detection/resolution in air traffic management. In *36th IEEE Conference on Decision and Control*.

Dowek, G., C. Muñoz, and A. Geser. 2001. Tactical conflict detection and resolution in 3-D airspace. In *4th USA/Europe Air Traffic Management R&D Seminar (ATM-2001)*. Santa Fe, New Mexico. Extendeed version available as ICASE Report No. 2002-12 NASA/CR-2002-211637.

Durand, N., J.-M. Alliot, and F. Medioni. 2000. Neural nets trained by genetic algorithms for collision avoidance. In *Applied Artificial Intelligence*, Number 3.

Durand, N., J.-M. Alliot, and J. Noailles. 1996. Automatic aircraft conflict resolution using genetic algorithms. In *Symposium on Applied Computing*. Philadelphia, PA.

Eby, M. 1994. A self-organizational approach for resolving air traffic conflicts. *Lincoln Laboratory Journal* 7 (2): 239–254.

Frazzoli, E., Z.-H. Mao, J.-H. Oh, and E. Feron. 2001. Resolution of conflicts involving many aircraft via semidefinite programming. *Journal of Guidance, Control, and Dynamics* 24 (1): 79–86.

Geser, A., and C. Muñoz. 2002. A geometric approach

to strategic conflict detection and resolution. Proceedings of the 21st Digital Avionics Systems Conference.

Geser, A., C. Muñoz, G. Dowek, and F. Kirchner. 2002, May. Air traffic conflict resolution and recovery. Technical Report ICASE Report No. 2002-12 NASA/CR-2002-211637, ICASE-NASA Langley, ICASE Mail Stop 132C, NASA Langley Research Center, Hampton VA 23681-2199, USA.

Granger, G., N. Durand, and J.-M. Alliot. 2001. Optimal resolution of en route conflicts. In *4th USA/Europe Air Traffic Management R&D Seminar (ATM-2001)*. Santa Fe, New Mexico.

Hoekstra, J., R. Ruigrok, R. van Gent, J. Visser, B. Gijsbers, M. Valenti, W. Heesbeen, B. Hilburn, J. Groeneweg, and F. Bussink. 2000, May. Overview of NLR free flight project 1997-1999. Technical Report NLR-CR-2000-227, National Aerospace Laboratory (NLR).

Kuchar, J., and L. Yang. 2000, December. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems* 1 (4): 179–189.

McDonald, J., and R. Vivona. 2000, November. Strategic airborne conflict detection of air traffic and area hazards. Technical Report NASA Contract: NAS2-98005 RTO-29, TITAN Systems Corporation, SRC Division.

NASA 1999. Concept definition for Distributed Air/Ground Traffic Management (DAG-TM), version 1.0. Advanced Air Transportation Technologies (AATT) Project. NASA Ames Research Center. NASA Langley Research Center.

Owre, S., J. M. Rushby, and N. Shankar. 1992, June. PVS: A prototype verification system. In *11th International Conference on Automated Deduction (CADE)*, ed. D. Kapur, Volume 607 of *Lecture Notes in Artificial Intelligence*, 748–752. Saratoga, NY: Springer-Verlag.

RTCA 1995. Final report of the RTCA board of directors' select committee on free flight. Technical Report Issued 1-18-95, RTCA, Washington, DC.

Sanford, B., K. Harwood, S. Nowlin, H. Bergeron, H. Heinrichs, G. Wells, and M. Hart. 1993, October. Center/TRACON automation system: Development and evaluation in the field. In *38th Annual Air Traffic Control Association Conference*.

Tomlin, C., G. Pappas, and S. Sastry. 1998. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control* (4).

Wing, D., R. Adams, B. Barmore, and D. Moses. 2001. Airborne use of traffic intent information in a distributed air-ground traffic management concept: Experiment design and preliminary results. In *4th USA/Europe Air Traffic Management R&D Seminar (ATM-2001)*. Santa Fe, New Mexico.

AUTHOR BIOGRAPHIES

RICKY BUTLER is a senior research engineer at the NASA Langley Research Center. He has led the formal methods program at Langley since 1989. The major goals of the Langley formal methods program are to advance the state of the art in formal methods, make it practical for use on high integrity systems, and to orchestrate the transfer of this technology to U.S. industry through carefully designed demonstration projects. The formal methods team consists of seven NASA civil servants and three researchers at the new National Institute of Aerospace near Langley. Mr. Butler won the 1991 and 1996 H.J.E. Reid Awards (Langley's best research paper award). He received NASA's Exceptional Achievement Medal in 1997. His research interests include formal methods, fault-tolerance, and reliability analysis.

JEFFREY MADDALON is a research engineer at the NASA Langley Research Center. He works on the formal verification of both air traffic management algorithms and fault-tolerant computer architectures. He has also worked in areas of flight simulation, real-time computing, embedded systems, and avionics systems. He received a bachelor's of science degree in computer engineering from Virginia Tech and a master's of science degree in computer science from the College of William and Mary.

ALFONS GESER is interested in inductive theorem proving, term rewriting, and model checking, and their practical application. He worked as a researcher at the Universities of Ulm, Passau, and Tuebingen, and as an ASIC Design Engineer at a hardware design company in Passau, Germany. He joined the National Institute of Aerospace (formerly ICASE) as a Senior Staff Scientist in Jan 2001. Currently he is involved in three NASA Langley formal verification projects. His e-mail address is <geser@nianet.org> and his web page is <research.nianet.org/~geser>.

CESAR A. MUNOZ is a senior staff scientist at the National Institute of Aerospace. He received his Ph.D. in Computer Science from the University of Paris 7 in 1997. After completing his Ph.D., he spent one and a half years as an International Fellow in the Formal Methods Group of the Computer Science Laboratory at SRI International in Menlo Park, CA. He joined ICASE at Langley Research Center in May 1999. Since January

2002 he is leading the Formal Methods group at the National Institute of Aerospace in Hampton, VA. His e-mail address is <munoz@nianet.org> and his web page is <research.nianet.org/~munoz>.